Journal of Scientific Computing, Vol. 10, No. 2, 1995

Parallel Solution of Tridiagonal Systems for the Poisson Equation

U. Schumann[1] and M. Strietzel[1]

Received November 1, 1994

A method is described to solve the systems of tridiagonal linear equations that result from discrete approximations of the Poisson or Helmholtz equation with either periodic, Dirichlet, Neumann, or shear-periodic boundary conditions. The problem is partitioned into a set of smaller Dirichlet problems which can be solved simultaneously on parallel or vector computers leaving a smaller tridiagonal system to be solved on one of the processors.

KEY WORDS: Tridiagonal linear systems; parallel computers; vector computers; Helmholtz equation; Poisson equation.

1. INTRODUCTION

For numerical simulation of incompressible fluid flows one has to solve the Poisson equation for pressure once for each integration time step, see, e.g., Gerz *et al.* (1989). By means of Fourier transformation a three-dimensional or two-dimensional Poisson equation gets reduced to a system of one-dimensional Helmholtz equations

$$p'' - \lambda^2 p = q, \qquad \text{on} \quad 0 < x \leqslant L \tag{1.1}$$

which have to be solved for each Fourier mode. The Helmholtz parameter λ^2 is related to the vector sum of the wave numbers of the Fourier wave functions in the original multi-dimensional problem. It is either zero or positive, depending on the boundary conditions in the transformed directions and on the wave-number magnitude [Schumann and Sweet (1988)].

[1] DLR, Institut für Physik der Atmosphäre, Postfach 1116, 82230 Oberpfaffenhofen, Germany. E-mail: Ulrich.Schumann@dlr.de.

854/10/2-1

Here, we consider the one-dimensional Helmholtz equation, first with periodic boundary conditions

$$p(0) = p(L), \qquad p'(0) = p'(L) \tag{1.2}$$

Dirichlet ($p(0) = 0$, $p(L) = 0$) and Neumann ($p'(0) = 0$, $p'(L) = 0$) boundary conditions will be discussed in Section 5 of this paper, together with the so-called "shear-periodic" boundary conditions which we need to simulate sheared turbulent flows [Schumann (1985) and Gerz *et al.* (1989)].

In order to make the pressure solution consistent with the discretized equations of fluid motion, one has to solve the discretized form of the Helmholtz equation: On an equidistant grid with N grid points $x_i = i\,\Delta x$, $\Delta x = L/N$, $i = 1, 2, ..., N$, the discrete solutions $p_i \cong p(x_i)$ and given sources $q_i = q(x_i)\,\Delta x^2$ satisfy the tridiagonal linear system of equations

$$p_{i-1} - a p_i + p_{i+1} = q_i, \qquad i = 1, 2, ..., N \tag{1.3}$$

with $a = 2 + \lambda^2\,\Delta x^2$ and periodic boundary conditions $p_0 = p_N$, $p_{N+1} = p_1$. The system is singular for $\lambda = 0$ ($a = 2$), in which case a solution, which is determined up to an additive constant, exists only if the sum of all sources vanishes, $\sum_1^N q_i = 0$, for consistency with the discrete Poisson operator.

On a single processor, this system can be solved using Gaussian elimination. Such a solver requires an order $7N$ multiplications, $1N$ divisions, and $6N$ additions or subtractions. When the computing time per multiplication equals that for additions (A), and is half that for divisions, then the computing time increases with $t_1 = 15AN$. These computations often form a major part of the whole integration effort.

Being an elliptical equation, the Helmholtz equation requires to relate the solution at one point to all other grid points within the solution domain. This contrasts with the remainder of the equations of motion which are essentially parabolic or hyperbolic and can be approximated by discrete systems which contain only local interactions. For parallel treatment of the equations of motion, methods which rely on domain decomposition are well suited therefore. Hence, we seek for a corresponding algorithm to solve the Poisson or Helmholtz equation.

Methods to solve large tridiagonal linear systems on vector and parallel computers have been reviewed by Gallivan *et al.* (1990), Ortega (1988), van der Vorst (1987), and Bondeli (1991). The method to be described in this paper is similar to the partition method derived by Wang (1981) for diagonally dominant tridiagonal systems with variable coefficients. He divides the matrix of the linear system into blocks and first eliminates the unknowns from the block's interior by a suitable Gaussian elimination procedure. This results into a reduced set of linear equations

coupling the blocks. Wang's algorithm requires to perform vector operations of different lengths which needs special coding for parallel processing with a fixed number of processors. Variants suitable both for MIMD (multiple instruction-multiple data) and SIMD (single instruction-multiple data) computers has been deduced by Kowalik *et al.* (1984) and Bondeli (1991). These partition methods, also called "divide and conquer methods," have been described only for nonperiodic systems. Our method will be shown to be more efficient than these algorithms, mainly because we take advantage of the constant coefficients and of the symmetry of the given system. Moreover, we use a different algorithmic idea, which gives additional insight and leads to proper treatment of systems with zero or very large Helmholtz parameters.

The algorithm described in this paper is basically a discrete analogy to a "matching procedure" described by Israeli *et al.* (1993) to solve the continuous Helmholtz equation. They solve the continuous equation with zero Dirichlet boundary values on subdomains and then add properly weighted continuous solutions for the homogeneous system with unit boundary values at either side of any subdomain.

2. THE DOMAIN PARTITION

We assume that we can use m processors, $1 < m < N$, each with sufficient amount of local storage. We assume that the processors are arranged to allow for data exchange between all "node processors" and one "host processor" and also locally with the nearest neighbours along a ring. The algorithm can also be used to perform most of the computations in parallel on a vector computer with shared memory.

Let $n = N/m$ be an integer number. The index domain $1 \leqslant i \leqslant N$ is split into m equal subdomains denoted by $j = 1, 2, \ldots, m$, each containing n contiguous grid points. In each subdomain the unknowns and source values are

$$p_i^{(j)} = p_{(j-1)n+i}, \quad q_i^{(j)} = q_{(j-1)n+i}, \qquad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, m \tag{2.1}$$

They satisfy

$$p_{i-1}^{(j)} - a p_i^{(j)} + p_{i+1}^{(j)} = q_i^{(j)} \tag{2.2}$$

with interface conditions

$$p_0^{(j)} = p_n^{(j-1)}, \qquad p_{n+1}^{(j)} = p_1^{(j+1)} \tag{2.3}$$

and periodic boundary conditions

$$p_0^{(1)} = p_n^{(m)}, \qquad p_{n+1}^{(m)} = p_1^{(1)} \tag{2.4}$$

Each subdomain is assigned to a processor.

3. CONSTRUCTION OF THE SOLUTION METHOD

We first solve a localized problem on each subdomain $j = 1, 2, \ldots, m$:

$$\bar{p}_1^{(j)} = 0 \tag{3.1}$$

$$\bar{p}_{i-1}^{(j)} - a\bar{p}_i^{(j)} + \bar{p}_{i+1}^{(j)} = q_i^{(j)}, \qquad i = 2, 3, \ldots, n \tag{3.2}$$

No data exchange of boundary values is needed since $\bar{p}_{n+1}^{(j)} = \bar{p}_1^{(j+1)} = 0$. This intermediate solution satisfies the inhomogeneous equations everywhere except at the left boundaries of each subdomain. In order to construct the correct solution everywhere, we need the specific solution of the homogeneous system with unit boundary value at the left boundary of a subdomain:

$$r_1 = 1 \tag{3.3}$$

$$r_{i-1} - ar_i + r_{i+1} = 0, \qquad i = 2, 3, \ldots, n \tag{3.4}$$

$$r_{n+1} = 0 \tag{3.5}$$

The final solution is obtained by superposition

$$p_i^{(j)} = \bar{p}_i^{(j)} + R_j r_i + R_{j+1} r_{n+2-i} \tag{3.6}$$

By construction, this solution satisfies the given equations at all interior grid points $i = 2, 3, \ldots, n$ of each subdomain $j = 1, 2, \ldots, m$. The amplitudes R_j have to be determined such that

$$p_n^{(j-1)} - ap_1^{(j)} + p_2^{(j)} = q_1^{(j)} \tag{3.7}$$

which requires

$$\begin{aligned} &\bar{p}_n^{(j-1)} + R_{j-1} r_n + R_j r_2 - a[\bar{p}_1^{(j)} + R_j r_1 + R_{j+1} r_{n+1}] \\ &\quad + \bar{p}_2^{(j)} + R_j r_2 + R_{j+1} r_n = q_1^{(j)} \end{aligned} \tag{3.8}$$

Since $\bar{p}_1^{(j)} = 0$, $r_1 = 1$, $r_{n+1} = 0$, this simplifies to

$$r_n R_{j-1} - (a - 2r_2) R_j + r_n R_{j+1} = q_1^{(j)} - \bar{p}_n^{(j-1)} - \bar{p}_2^{(j)} \tag{3.9}$$

for $j=1, 2,..., m$. The equations with outer boundary conditions at $i=1$, $j=1$ and at $i=n, j=m$ are satisfied if

$$R_0 = R_m, \quad \bar{p}_n^{(0)} = \bar{p}_n^{(m)}, \qquad R_{m+1} = R_1 \tag{3.10}$$

We note that this is a periodic symmetric system of m tridiagonal linear equations with constant coefficients. The system becomes singular if $2r_n = a - 2r_2$.

The homogeneous system in Eq. (3.4) approximates $r'' - \lambda^2 r = 0$ with boundary conditions $r(0)=1$, $r(L/m)=0$. The solution of this continuous equation is

$$r(x) = 1 - \bar{x} \qquad \text{if} \quad \lambda = 0 \tag{3.11}$$

$$r(x) = \frac{\sinh[\bar{\lambda}(1-\bar{x})]}{\sinh(\bar{\lambda})} \qquad \text{if} \quad \lambda^2 > 0 \tag{3.12}$$

with $\bar{\lambda} = \lambda L/m$, $\bar{x} = xm/L$. Also the discrete system shows a linear solution if $\lambda = 0$, such that $r_n = 1/n$, $r_2 = 1 - r_n$. Hence, the periodic system in Eq. (3.9) is singular for $\lambda = 0$. It is also a consistent system in this case, since

$$\sum_{j=1}^{m} q_1^{(j)} - \bar{p}_n^{(j-1)} - \bar{p}_2^{(j)} = \sum_{j=1}^{m} q_1^{(j)} - \bar{p}_n^{(j)} - \bar{p}_2^{(j)} = \sum_{j=1}^{m} \sum_{i=1}^{n} q_i^{(j)} \tag{3.13}$$

vanishes as for the original system in Eq. (1.3). The latter relation can be verified by summing Eqs. (3.2) from $i=2$ to $i=n$, with $a=2$, using the boundary conditions of Eq. (3.1) at j and $j+1$.

For large Helmholtz parameters $\lambda \gg 1/L$, the off-diagonal coefficients become small relative to the diagonal ones, i.e., $2r_n \ll a - 2r_2 \cong \lambda^2 \Delta x^2$, in which case the solutions R_j may be obtained locally ignoring the coupling with the neighboring solutions $R_{j\pm1}$. The approximations $r_2 \cong r(1/n)$, $r_n \cong r(1-1/n)$, with $r(x)$ given in Eq. (3.12), may be used to estimate when $r_n \ll a/2 - r_2$.

4. THE ALGORITHM FOR PERIODIC BOUNDARY CONDITIONS

The solutions $p_i^{(j)}$ are obtained in three steps:

1. The r_i are determined according to Eqs. (3.4) together with the $\bar{p}_i^{(j)}$ from Eqs. (3.2) for given $q_i^{(j)}$:

   For each processor $j := 1$ to m DO
   $b_n := 1/a$; $\bar{p}_n^{(j)} := -b_n \times \bar{p}_n^{(j)}$;
   For $i := n-1$ down to 2 DO
   $b_i := 1/(a - b_{i+1})$; $\bar{p}_i^{(j)} := (\bar{p}_{i+1}^{(j)} - \bar{p}_i^{(j)}) \times b_i$;
   end DO

$r_2 := b_2$;
For $i := 3$ to n DO
$r_i := b_i \times r_{i-1}$; $\bar{p}_i^{(j)} := \bar{p}_i^{(j)} + b_i \times \bar{p}_{i-1}^{(j)}$;
end DO
end DO

Here it is assumed that the sources $q_i^{(j)}$ are stored initially in place of the variables $\bar{p}_i^{(j)}$. Note that $\bar{p}_1^{(j)}$ still contains the original source value $q_1^{(j)}$, as required for the second step. The b_i, $i = 1,..., n$ are used as temporary array.

For large n this step requires about $3n$ multiplications, n divisions and $3n$ additions/subtractions or a computing time $t_{\bar{p}} = 8An$, if A is the computing time per addition or multiplication, and $2A$ is the computing time per division. In principle, the r_i could be precomputed on one processor and then distributed to all processors, but this would not pay off (except on shared memory machines) because the local computation requires just n multiplications which commonly consume less computing time than the communication to all processors.

2. If the off-diagonal components of Eq. (3.9) are not small, i.e., if $2r_n/(a - 2r_2) > \varepsilon$, where $0 \leqslant \varepsilon \ll 1$ is a given error bound, then a global solution is required. In this case the results $\bar{p}_2^{(j)}$, $\bar{p}_n^{(j)}$, and the sources $q_1^{(j)}$ have to be transformed to the host processor where Eq. (3.9) is then to be solved for the R_j and from where R_j and R_{j+1} is transferred back to the individual processors. The algorithm for this part works as follows:

$c := 1/r_n$; $R_1 := (\bar{p}_1^{(1)} - \bar{p}_n^{(m)} - \bar{p}_2^{(1)}) \times c$;
For $j := 2$ to m DO $R_j := (\bar{p}_1^{(j)} - \bar{p}_n^{(j-1)} - \bar{p}_2^{(j)}) \times c$; end DO
$\bar{a} := (a - 2 \times r_2) \times c$; $d_1 := 1/\bar{a}$; $e_1 := d_1$; $R_1 := -R_1 \times d_1$; $f := -\bar{a}$;
$c := 1$;
For $j := 2$ to $m - 1$ DO
$z := 1/(\bar{a} - d_{j-1})$; $d_j := z$; $e_j := e_{j-1} \times z$; $R_j := (R_{j-1} - R_j) \times z$;
$R_m := R_m - R_{j-1} \times c$; $f := f + c \times e_{j-1}$; $c := c \times d_{j-1}$;
end DO
$d_{m-1} := d_{m-1} + e_{m-1}$; $c := 1 + c$;
IF $(\lambda^2 = 0)$ then $R_m := 0$; else $R_m := (R_m - c \times R_{m-1})/(f - c \times d_{m-1})$; end IF
$R_{m-1} := R_{m-1} + d_{m-1} \times R_m$;
For $j := m - 2$ down to 1 DO $R_j := R_j + d_j \times R_{j+1} + e_j \times R_m$; end DO
$R_{m+1} := R_1$;

Here, d_j, e_j, $j=1,\ldots,m$; c, f, z, and $\bar{a}$ are temporary variables. This step requires about $8m$ multiplications, m divisions, and $8m$ additions, or $t_h = 18Am$ as computing time on the host. In addition $5m$ words have to be transferred between the host and its node processors causing a communication time of order $t_C = 5Cm$, where C is the time needed per communicated word.

If $2r_n/(a-2r_2) \leqslant \varepsilon$, the R_j can be determined locally on each processor. For this purpose, the result $\bar{p}_n^{(j-1)}$ has to be transferred from processor $(j-1)$ to (j). The solution R_j has to be transferred backwards from processor (j) to $(j-1)$. This requires communication of two data values between each pair of nearest neighbors only.

3. Finally, we obtain the solution $p_i^{(j)}$ from Eq. (3.6):
For each processor $j := 1$ to m DO
$p_1^{(j)} := R_j$;
For $i := 2$ to n DO $p_i^{(j)} := \bar{p}_i^{(j)} + R_j \times r_i + R_{j+1} \times r_{n+2-i}$; end DO
end DO

The $p_i^{(j)}$ can be stored in the same places as $\bar{p}_i^{(j)}$ and $q_i^{(j)}$. This part requires $2n$ multiplications and $2n$ additions per processor, i.e., a computing time $t_p = 4An$.

In the standard case, with small ε, the total computation time t_m per solution of the Helmholtz equation with m parallel processors equals $t_m = t_{\bar{p}} + t_h + t_p + t_C = 12An + 18Am + 5Cm \equiv \alpha n + \gamma m$. This time reaches its minimum $t_{\text{opt}} = 2(\alpha\gamma N)^{1/2}$ for $m = m_{\text{opt}} = (\alpha N/\gamma)^{1/2}$. Hence, for $\alpha \cong \gamma$ and N in between 100 and 10000 (which is very large in view of the three-dimensional origin of the problem), the optimum number of processors lies in between 10 and 100. Slow communication increases γ and hence decreases the optimal number of processors. The speed-up factor $S = t_1/t_m$ of the parallel algorithm in comparison to the sequential algorithm is $S = (15Anm)/(12An + 18Am + 5Cm)$.

5. THE ALGORITHM FOR OTHER BOUNDARY CONDITIONS

The algorithm becomes simpler when we consider Dirichlet boundary conditions

$$p_0 = 0, \qquad p_{N+1} = 0 \tag{5.1}$$

at the left and right boundaries, where

$$R_0 = 0, \qquad \bar{p}_n^{(0)} = 0, \qquad R_{m+1} = 0 \tag{5.2}$$

is used instead of Eq. (3.10). This results in a purely (nonperiodic) tridiagonal system for the R_j which is nonsingular for $a \geqslant 2$. Hence, the computing time to solve the system on m processors is a little less than for the periodic system and amounts to $t_m = 12An + 10Am + 5Cm$. This value might be compared with $t_1 = 10AN$ for the sequential algorithm. The time t_m is smaller than the computing time $20An + 10Am + 6Cm$ of Kowalik's algorithm (1984). Also the algorithm of Wang (1981) requires more operations than ours. Bondeli (1991) does not give the computing time for parallel systems. His algorithm is derived for arbitrary positive definite tridiagonal systems and requires to solve three tridiagonal systems of rank n in step 1 and one tridiagonal system of rank $2m-2$ in step 2. Hence, our algorithm is more efficient than Bondeli's version for the special case of constant coefficients considered in this paper.

Neumann boundary conditions

$$p_0 = p_1, \qquad p_{N+1} = p_N \tag{5.3}$$

can also be implemented using

$$r_n R_0 = (1 - r_2)\, R_1, \qquad \bar{p}_n^{(0)} = 0 \tag{5.4}$$

for the left boundary and

$$R_m((a-1)\, r_n - r_{n+1}) + R_{m+1}((a-1)\, r_2 - r_3) = \bar{p}_n^{(m)} \tag{5.5}$$

for the right boundary instead of Eq. (3.10). This results also in a purely tridiagonal system for the R_j. It is singular for $a = 2$ if Neumann boundary conditions apply at both sides. The computing time is roughly (up to order A) the same as for the Dirichlet system.

Shear-periodic boundary conditions prescribe periodicity of complex Fourier modes p_i with a complex phase-shift factor W of unit magnitude, depending on shear and wave number [Schumann (1985)], so that

$$p_0 = W p_N, \qquad p_{N+1} = W^{-1} p_1 \tag{5.6}$$

This can be taken into account by setting

$$R_0 = W R_m, \qquad \bar{p}_n^{(0)} = W \bar{p}_n^{(m)}, \qquad R_{m+1} = W^{-1} R_1 \tag{5.7}$$

instead of Eq. (3.10). The resultant tridiagonal system for the R_j is of the periodic structure and is singular for $a = 2$. It requires a special coding for efficient treatment of the complex arithmetic [Schumann (1985)].

6. DISCUSSION AND CONCLUSION

We have tested the algorithms comparing the solutions to a reference solution, given by random numbers (between -0.5 and 0.5). All computations have been carried out in floating point arithmetic with a relative accuracy of $2^{-52} \cong 2.2 \cdot 10^{-15}$. For $m = 16$, $n = 64$, $L = 1$, and $\lambda = 0$, the maximum deviation between the solutions and the reference solution is computed for the periodic case to be $0.7 \cdot 10^{-12}$, and the maximum residual with respect to the source values is $0.8 \cdot 10^{-14}$. The errors decrease with increasing λ. Also, the nonsingular Dirichlet system gives somewhat smaller errors ($0.6 \cdot 10^{-12}$, $0.8 \cdot 10^{-15}$, respectively, for $\lambda = 0$).

The parallelized algorithm for the periodic system leads to slightly (about factor 4) more accurate results than does the standard sequential algorithm. A similar experience has been reported by van der Vorst (1987). This may be caused by the fact that the systems to be solved on the subdomains use Dirichlet boundary conditions and are nonsingular, therefore, even when the original problem is singular. Moreover, for $\lambda^2 > 0$ the final system, Eq. (3.9), is more strongly diagonally dominant than the original system of Eqs. (1.3).

We tested the algorithm for three-dimensional problems with equal numbers of grid points in each coordinate. The parallel Poisson solver was implemented on a transputer based Parsytec GCel-1024 parallel computer utilizing 4, 8 and 16 of its T800 processors. For portable coding ParMacs was chosen as the message passing system which is available on most parallel computer systems, at least in Europe. The small amount of local-memory on each transputer (4 MB) limits the resolution to 64^3 gridpoints on 4 transputers and 128^3 gridpoints on 8 and 16 nodes. The efficiency of the parallel program is an increasing function on the rank of the system in Eq. (3.2) denoted by n. A small ratio m/n reduces the time for solving the system in Eq. (3.9), which is the only sequential part of the solver, relative to the time used for solving the systems in Eq. (3.2) in parallel. This is an essential requirement for a good efficiency of the algorithm. On 4 processors the smallest ratio between m ($=4$) and n (up to 16) achievable was 1/4 and for this run we found a speed-up of 3.97 for the three-dimensional Poisson solver (including the time for fast Fourier transforms in two dimensions). This is an efficiency of 99%. For 8 processors and a resolution of 128^3 points (ratio $= 1/2$) the speed-up is 7.60 (95%). On 16 nodes the smallest ratio we could realize is $16/8 = 2$. For this ratio the speed-up is 14.11 (88%), which is reasonable for this problem size.

Note that the variants for all boundary conditions result in identical algorithms for steps 1 and 3, with the same instructions for all processors. Boundary conditions affect only step 2, i.e., the computations on the host

processor. Hence, the algorithm is applicable both on MIMD and SIMD computers.

In conclusion, a method has been described to solve the systems of tridiagonal linear equations resulting from the Poisson or Helmholtz equation on a regular grid in an efficient manner on parallel computers. In a future paper we will report on results obtained using this algorithm for turbulence simulations.

REFERENCES

Bondeli, S. (1991). Divide and conquer: A parallel algorithm for the solution of a tridiagonal linear system of equations, *Parallel Comput.* **17**, 435–442.

Gallivan, K. A., Plemmons, R. J., and Sameh, A. H. (1990). Parallel algorithms for dense linear algebra computations, *SIAM Rev.* **32**, 54–135.

Gerz, T., Schumann, U., and Elghobashi, S. E. (1989). Direct numerical simulation of stratified homogeneous turbulent shear flows, *J. Fluid Mech.* **200**, 563–594.

Israeli, M., Vozovoi, L., and Averbuch, A. (1993). Parallelizing implicit algorithms for time-dependent problems by parabolic domain decomposition, *J. Sci. Comp.* **8**, 151–166.

Kowalik, J. S., Lord, R. E., and Kumar, S. P. (1984). Design and performance of algorithms for MIMD parallel computers, in J. S. Kowalik (ed.), *High-Speed Computation*, Springer, Berlin, pp. 257–276.

Ortega, J. M. (1988). *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, pp. 108–131.

Schumann, U. (1985). Algorithms for direct numerical simulation of shear-periodic turbulence, in *Lecture Notes in Physics* **218**, Springer, Berlin, pp. 492–496.

Schumann, U., and Sweet, R. A. (1988). Fast Fourier transforms for direct solution of Poisson's equation with staggered boundary conditions, *J. Comput. Phys.* **75**, 123–137.

van der Vorst, H. A. (1987). Large tridiagonal and block tridiagonal linear systems on vector and parallel computers, *Parallel Computing* **5**, 45–54.

Wang, H. H. (1981). A parallel method for tridiagonal equations, *ACM Transact. Math. Software* **7**, 170–183.